



Deby - Reproducible and Maintainable Embedded Linux Environment with Poky

Kazuhiro Hayashi

CE Workgroup, The Linux Foundation (TOSHIBA Corporation)

Embedded Linux Conference Europe 2016

Oct 12, 2016



About this project

- **Shared Embedded Linux Distribution Project**
 - One of the activities of CEWG project
 - Goals: Create an industry-supported distribution of embedded Linux and provide support for long term
- **For more information about this project**
 - Shared Embedded Linux Distribution
 - http://elinux.org/Shared_Embedded_Linux_Distribution
 - CE Workgroup Linux Foundation
 - <http://www.linuxfoundation.org/collaborate/workgroups/celf>



Motivation

- **Linux is running on many kind of embedded systems**
 - Including the systems in civil infrastructure
- **Things to be considered to choose a base distribution**
 - The number of supported packages
 - Package versions
 - Supported hardware
 - Stability, number of bugs were fixed
 - The frequency of security updates and supported timespan
 - How to compile and customize packages



In our case

- **What we want to do**
 - Make custom embedded Linux environments
- **What we need**
 - Wider hardware support
 - Stability
 - Well tested packages are required
 - Many embedded developer are still want to use stable version
 - Long-term support
 - Over 10 years support required, especially for security fixes
 - (This is what we would like to contribute something)
 - Fully customizable build system



Our solution

Yocto Project "poky"

- One of the most popular reference distributions for embedded Linux
- Fully customizable build system
- Supports numerous embedded boards including modern ones
- Can be extended by meta-layer

Debian GNU/Linux

- Support many kind of CPUs: x86, ARM, PowerPC, MIPS (32bit/64bit)
- Release a stable version after two years of testing
- Long-term support for 5 years by Debian-LTS project



Our solution

Yocto Project "poky"

- One of the most popular reference distributions for embedded Linux
- Fully customizable build system
- Supports numerous embedded boards including modern ones
- Can be extended by meta-layer

Debian GNU/Linux

- Support many kind of CPUs: x86, ARM, PowerPC, MIPS (32bit/64bit)
- Release a stable version after two years of testing
- Long-term support for 5 years by Debian-LTS project

meta-debian

Deby



Definitions of the terms

- **meta-debian**

- A meta layer for the poky build system
 - Completely separated from OpenEmbedded-Core and other layers
- Allows cross-building Linux images using Debian source packages
- Source code
 - <https://github.com/meta-debian/meta-debian.git>

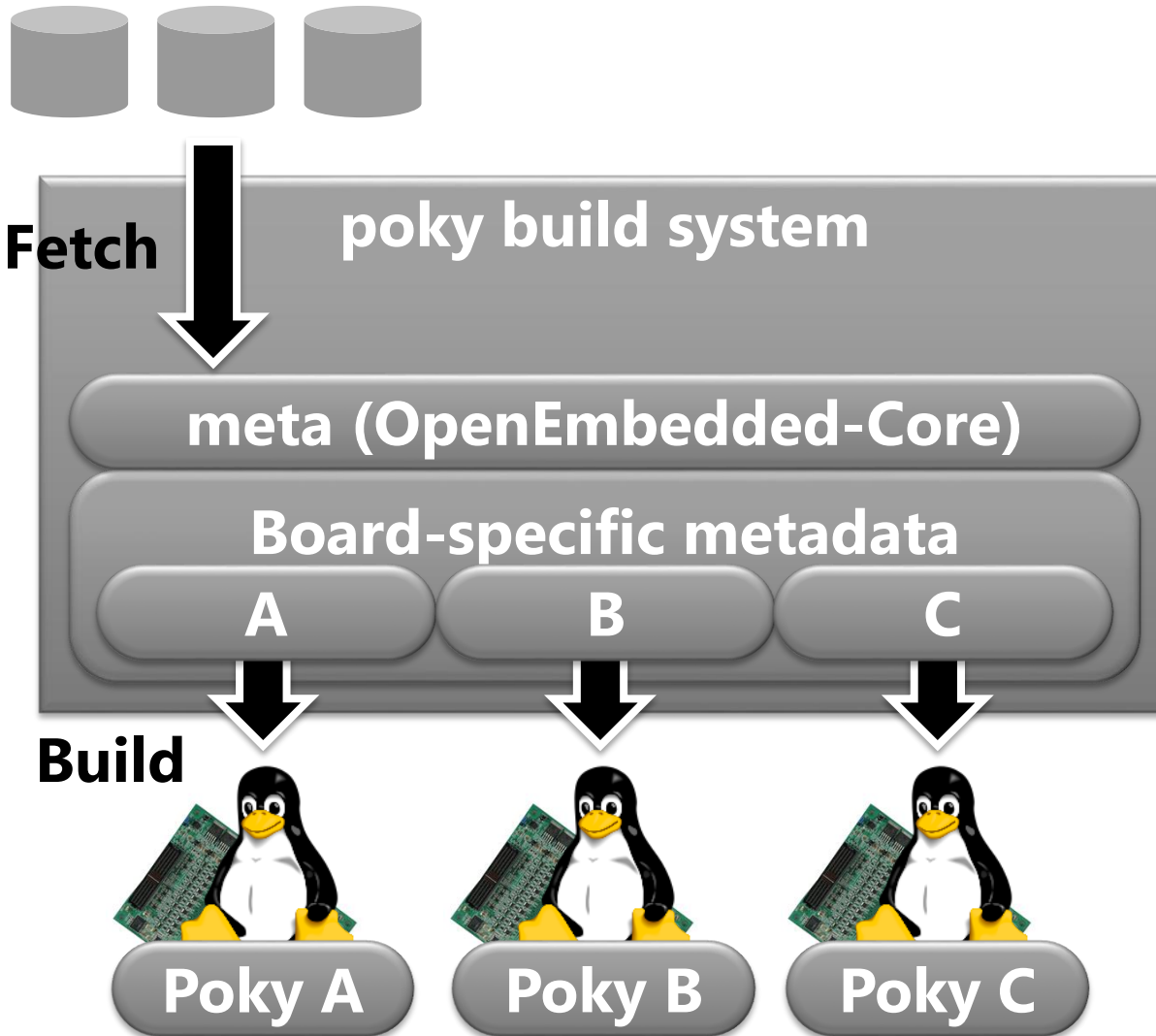
- **Deby**

- A reference distribution built with poky+meta-debian
- **Deby** = **Deb**ian + poky
- **Deby** = **Deb**ian-**like**
 - Cross-built from Debian source, but not same as Debian binary



Build system structure (poky)

Upstream source code

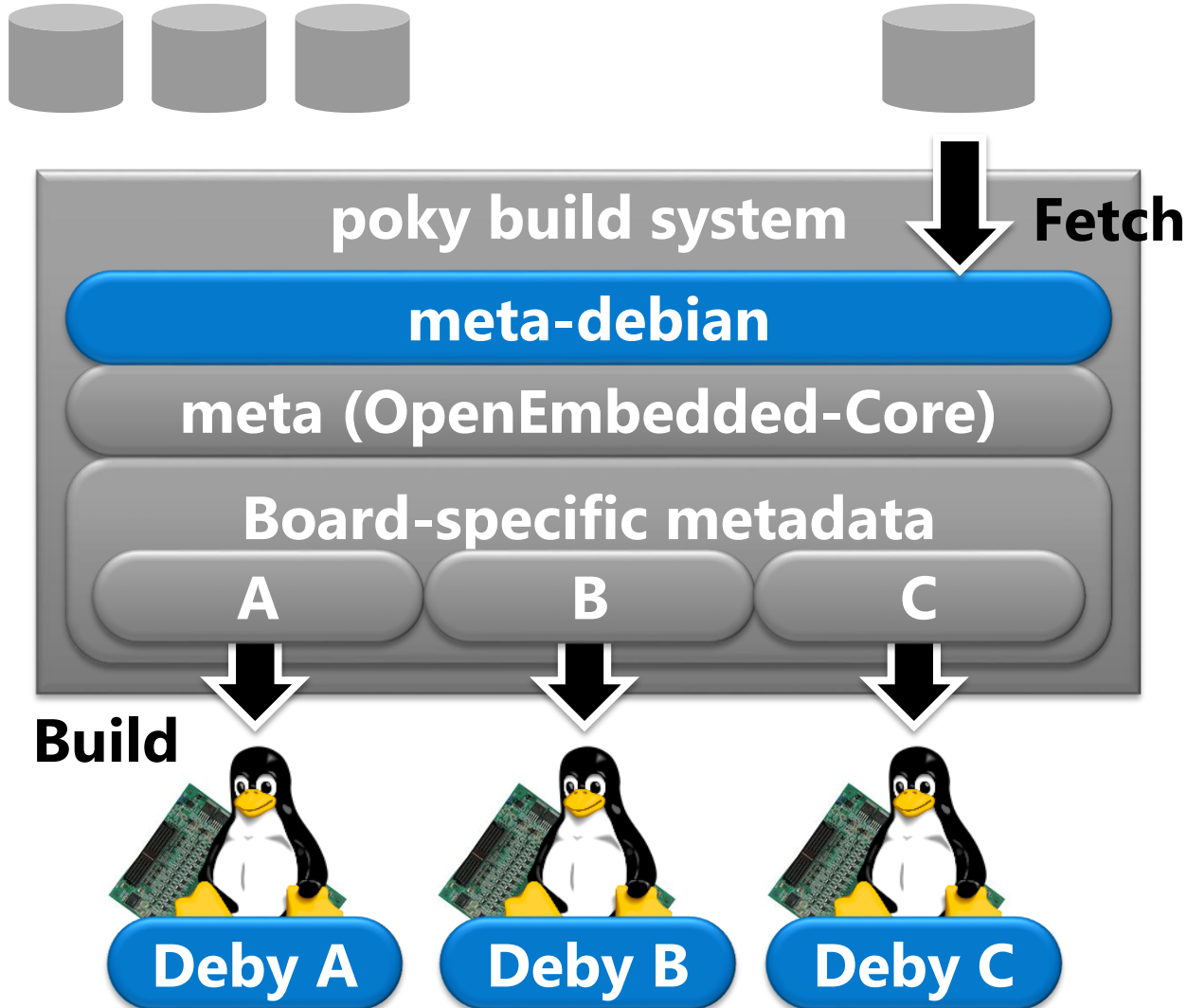




Build system structure (poky + meta-debian)

Upstream source code

Debian source packages

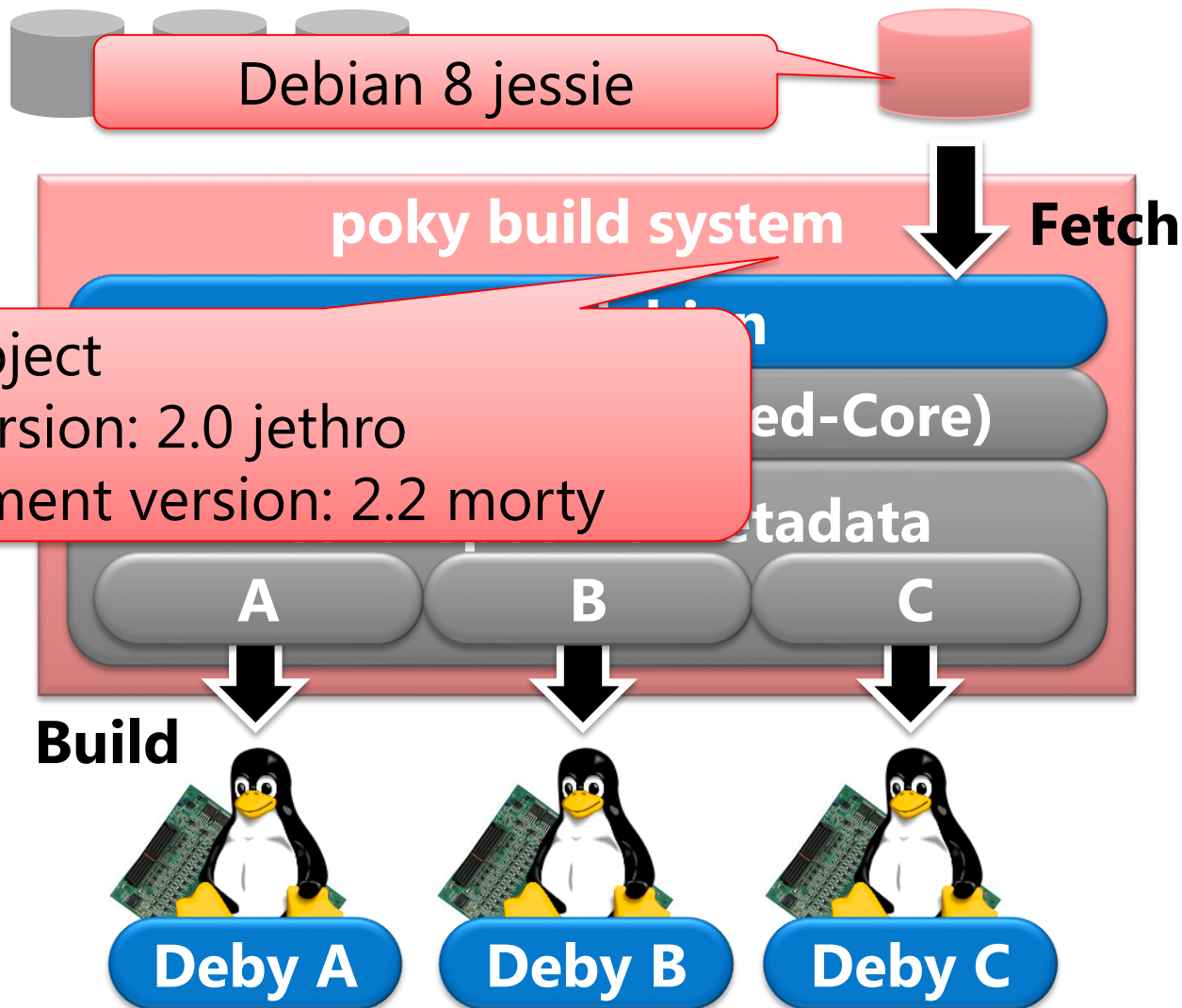




Target versions of Deby

Upstream source code

Debian source packages





Purpose of Deby

- **Create embedded Linux environments with**

- Wide embedded CPU support

- Stability

- Long-term support

- Fully customizable build system

With Debian stable release + LTS

With poky build system

- **Contribute to upstream**

- Debian, Debian LTS, and Yocto Project



Development policies of Deby

- **Follow Debian's packaging (debian/rules)**
 - Use the same configure/compile commands and options, install paths, binary package name, and dependencies as Debian
- **Add patches for supporting cross-compile**
 - Usually imported from OE-Core
- **Customize for embedded system if necessary**
 - Remove unneeded features, dependencies and packages
 - Ex: udeb packages for Debian installer
- **See also**
 - http://events.linuxfoundation.org/sites/events/files/slides/LinuxCon2015_meta-debian_r7.pdf



Quick start

- 1. Download the build tools**
 - 2. Setup build directory**
 - 3. Build minimal Linux image**
 - 4. Run minimal Linux image on QEMU**

 - 5. Build & install minimal SDK**
 - 6. Build application with SDK**
 - 7. Run application on QEMU**
- **See also meta-debian/README.md**
 - <https://github.com/meta-debian/meta-debian/blob/jethro/README.md>



Download build tools

- **Download poky**

```
$ git clone git://git.yoctoproject.org/poky.git  
$ cd poky  
$ git checkout jethro
```

- **Download meta-debian into the poky directory**

```
$ cd poky  
$ git clone https://github.com/meta-debian/meta-debian.git  
$ cd meta-debian  
$ git checkout jethro
```

• ← **meta-debian specific step**



Setup build directory

• Change the default configuration

- Enable meta-debian layer
- Enable "deby" distro (DISTRO = "deby")
- The default target machine is "qemux86" (MACHINE = "qemux86")
- TEMPLATECONF is used by oe-init-build-env script

```
$ export TEMPLATECONF=meta-debian/conf
```

• Run startup script

- This setup a build directory and environment variables automatically
- (builddir): name of build directory (optional)

```
$ source /path/to/poky/oe-init-build-env (builddir)
```



Build minimal Linux image

- **Run bitbake**

```
$ bitbake core-image-minimal
```

- **Built images (case of qemu86)**

- Output directly
 - /path/to/build/tmp/deploy/images/qemu86
- Kernel
 - bzImage-qemu86.bin
- Root filesystem
 - core-image-minimal-qemu86.ext4
 - core-image-minimal-qemu86.tar.gz



Run minimal Linux image on QEMU

- **Run built images on QEMU environment**

- `qemux86` / `qemux86-64` / `qemuppc` / `qemumips`

```
$ runqemu qemux86 nographic
```

```
$ runqemu qemux86-64 nographic
```

```
$ runqemu qemuppc nographic
```

```
$ runqemu qemumips nographic
```

- `qemuarm`

```
$ runqemu qemuarm nographic bootparams="console=ttyAMA0"
```



Build & install minimal SDK

- **Run bitbake**

```
$ bitbake meta-toolchain
```

- **Output (Host: x86_64, Target: qemux86)**

- /path/to/builddir/tmp/deploy/sdk/qemux86/deby-glibc-x86_64-meta-toolchain-i586-toolchain-8.0.sh
 - Self-extracting script

- **Install SDK to host environment**

```
$ sh deby-glibc-x86_64-meta-toolchain-i586-toolchain-8.0.sh
```



Build application with SDK

- **Create hello.c and Makefile**

```
/* hello.c */
#include <stdio.h>
int main(int argc, char **argv)
{
    printf("hello world\n");
    return 0;
}
```

```
# Makefile

hello: hello.o
```

- **Export SDK environment variables and make**

```
$ source /opt/deby/8.0/environment-setup-i586-deby-linux
$ make
```

- **See also Yocto Project Application Developer's Guide**

- <http://www.yoctoproject.org/docs/2.0/adt-manual/adt-manual.html#using-the-command-line>



Run application on QEMU

- **Copy hello to the filesystem image**

```
$ cd /path/to/builddir/tmp/deploy/images/qemux86
$ sudo mount -o loop ¥
  core-image-minimal-qemux86.ext4 /mnt
$ sudo cp /path/to/hello /mnt
$ sudo umount /mnt
```

- **Run application on QEMU**

```
$ runqemu qemux86 nographic
...
192.168.7.2 login: root
# /hello
hello world
```



New features

- **Supported Yocto Project version**
 - 2.0 jethro (Stable)
 - 2.2 morty (Development)
- **Kernel**
 - 4.4 LTS
 - 4.1 LTSI
- **The number of available recipes**
 - Approx. 500
- **Newly supported target machine**
 - BeagleBoard, PandaBoard



New features

- **Package management**
 - Run-time dpkg / apt
- **Tag based source code fetch and build**
 - Rebuild the Linux image that was built at the specific time
- **Summary generation**
 - Generate summary information of packages included in rootfs and SDK



Package management

- **This feature is available in OE-Core**
- **How to enable package management feature**

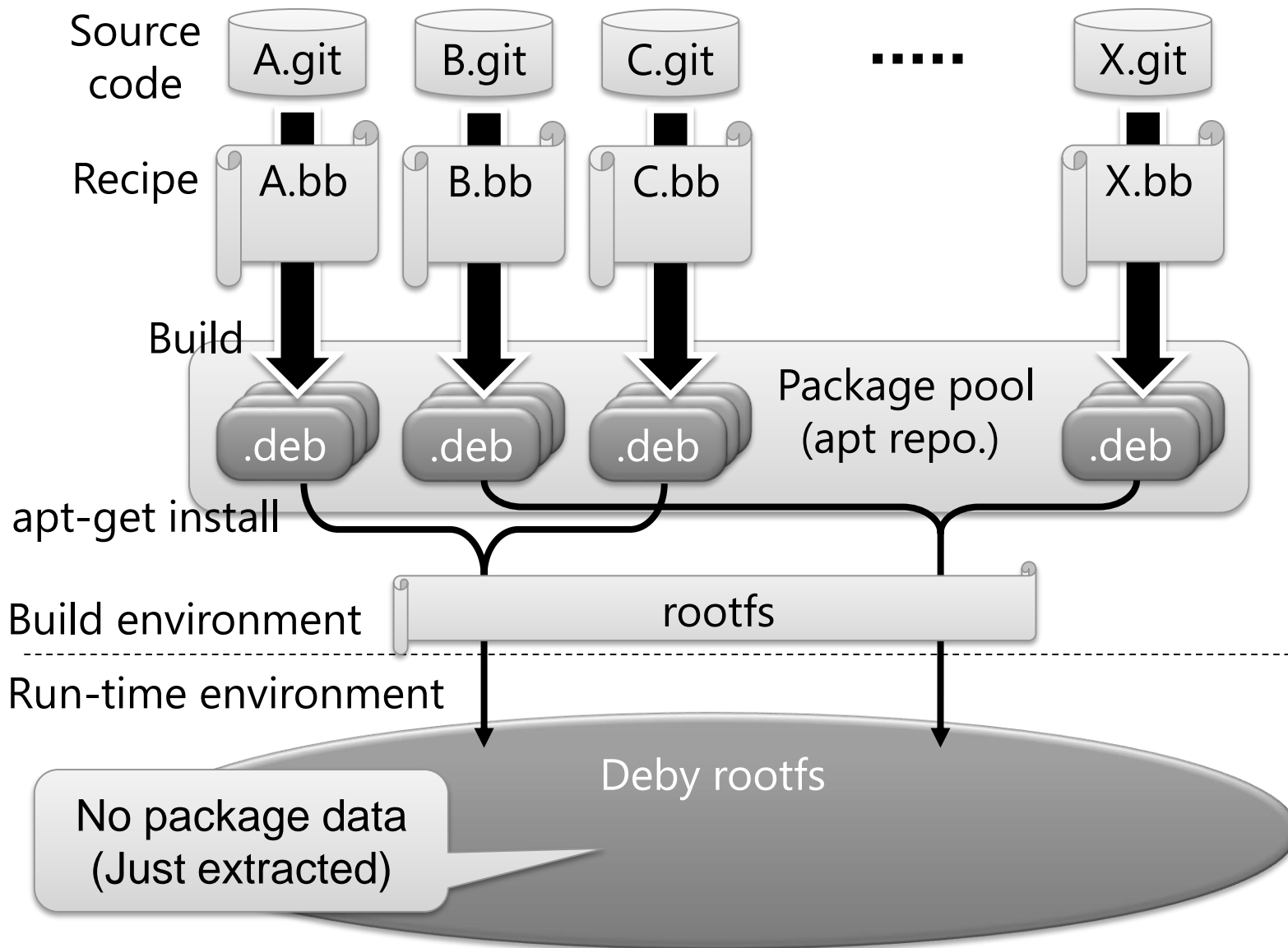
- Package management feature is disabled by default
- Add the following definition into local.conf

```
EXTRA_IMAGE_FEATURES += "package-management"
```

- **With package management feature, we can...**
 - Add binary packages into run-time environment
 - Temporally install/uninstall packages for system evaluation
 - Temporally install -dbg packages for debugging
 - Upgrade packages without stopping system
 - Install / upgrade packages without building & installing rootfs again

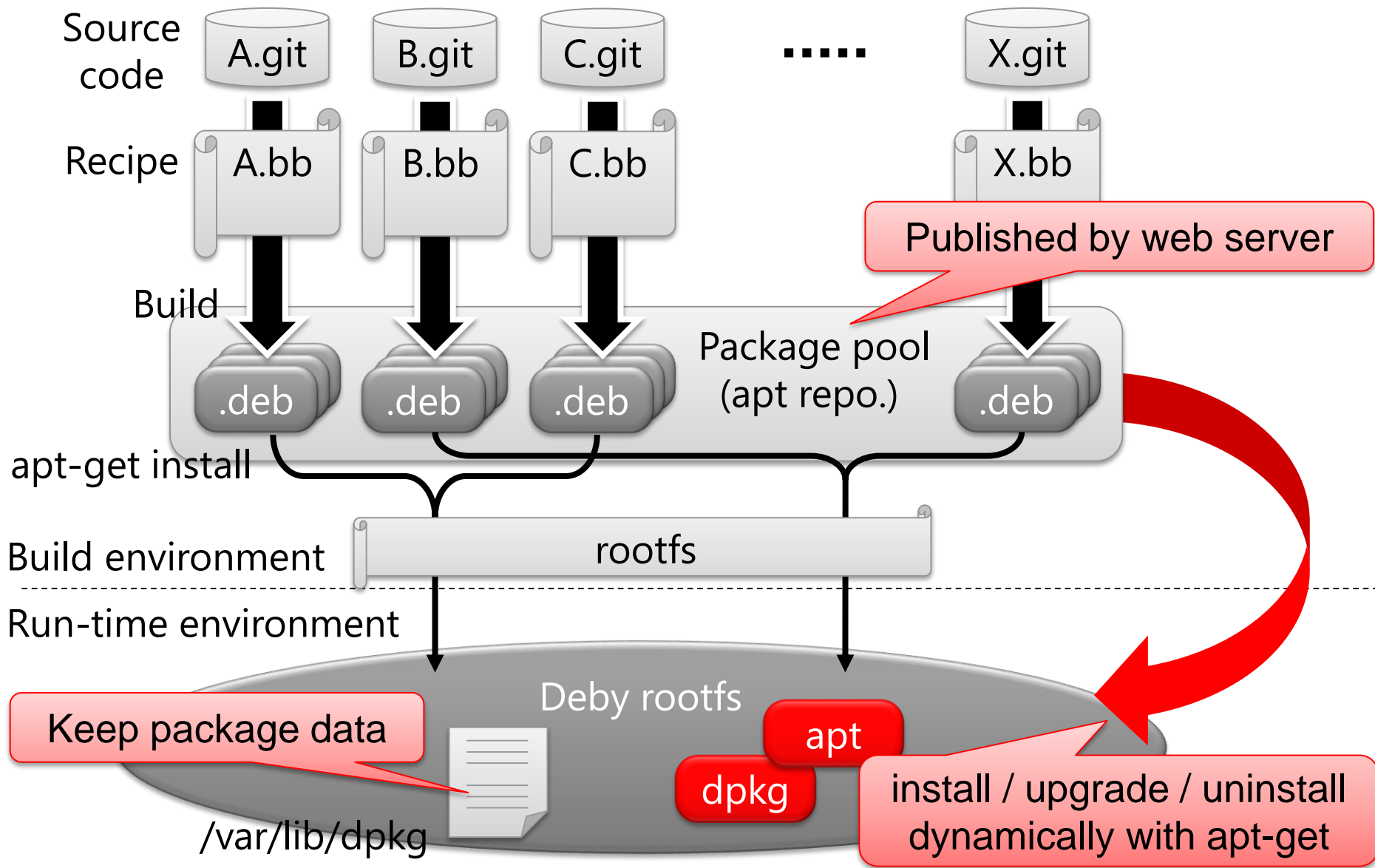


rootfs without package management





rootfs with package management



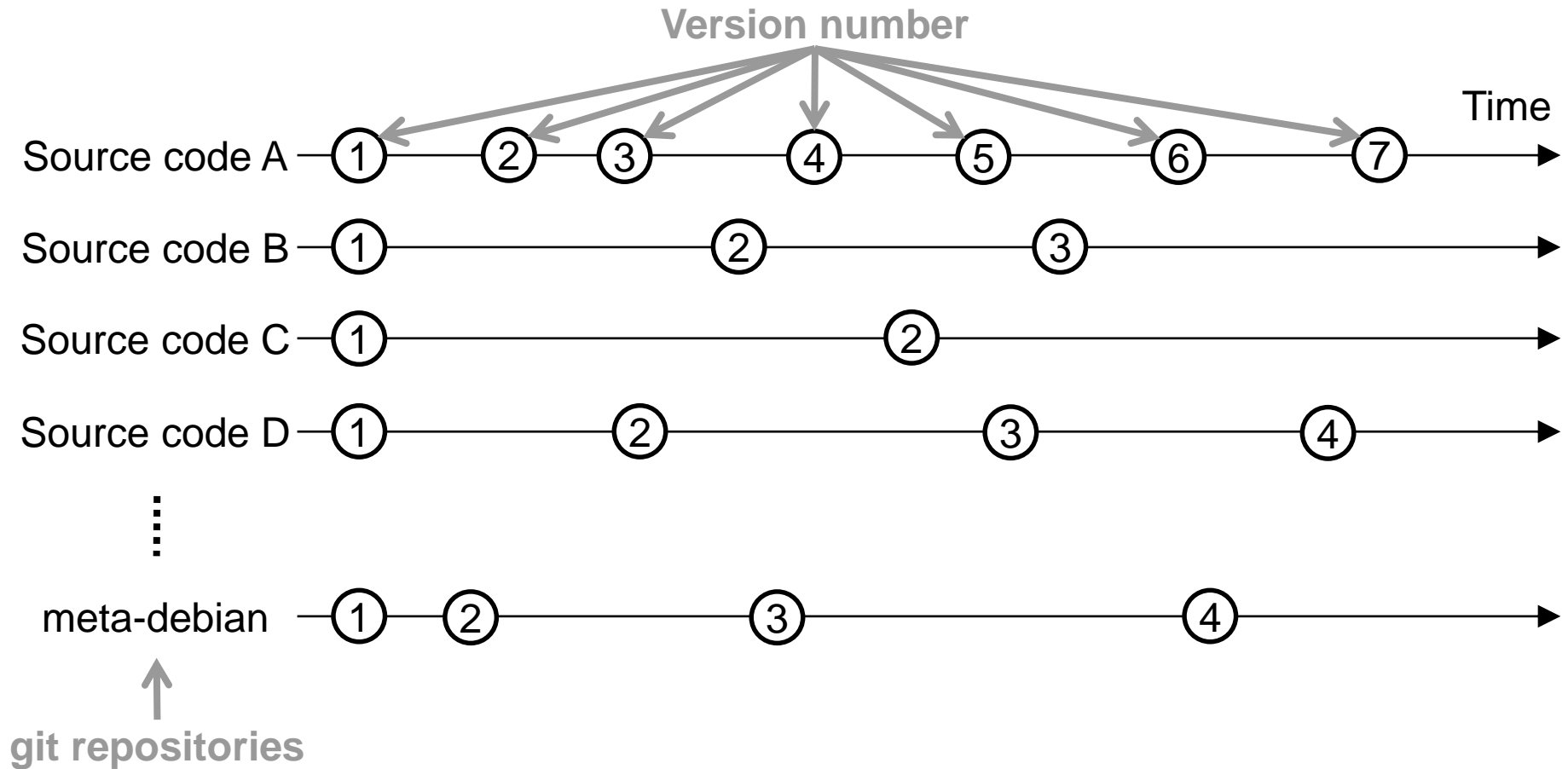


Tag based source code fetch and build

- **Issues in the default behavior of meta-debian**
 - No reproducibility
 - Cannot reproduce rootfs/SDK that was built at the specific time
 - Recipes always fetches the latest source code (the latest git commit)
 - To automatically import all security updates
- **Reproducible build**
 - One of the essential features in long-term maintenance
 - Useful for finding the source of issue in the old released image
- **Solution**
 - STEP1: Register a release tag in git repositories every release
 - STEP2: Reproduce an old release image by specifying a tag name
 - Add a new global variable: **GIT_REBUILD_TAG**

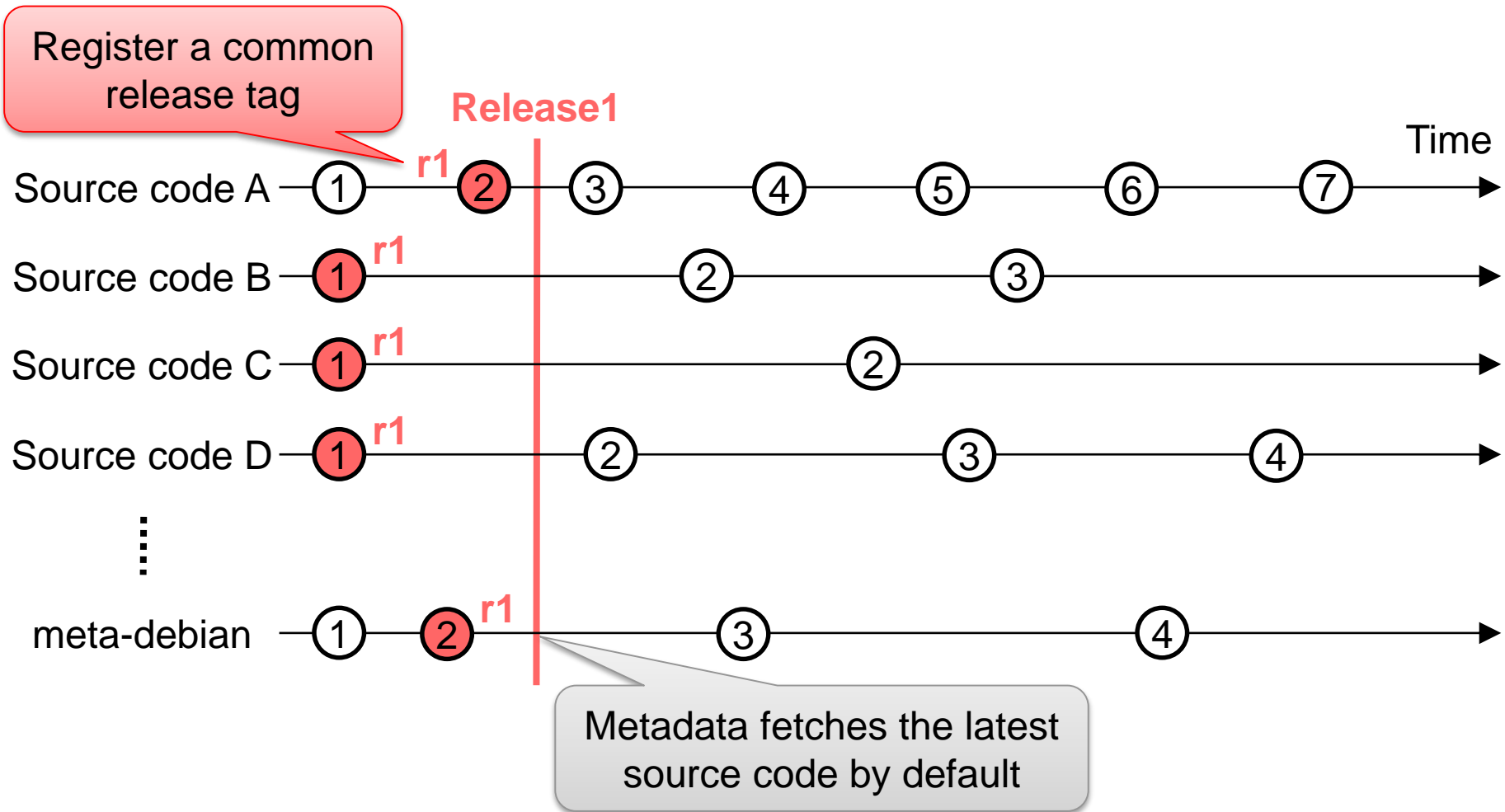


STEP1: Register a release tag



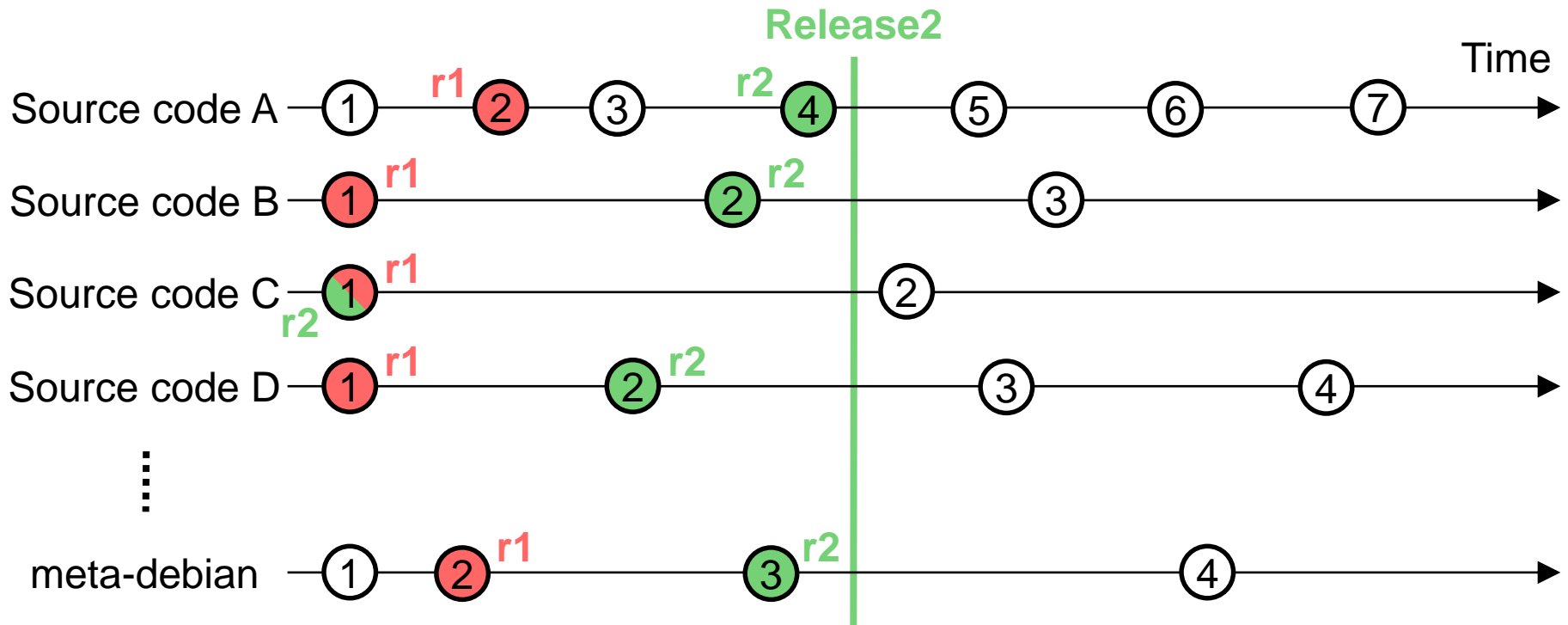


STEP1: Register a release tag



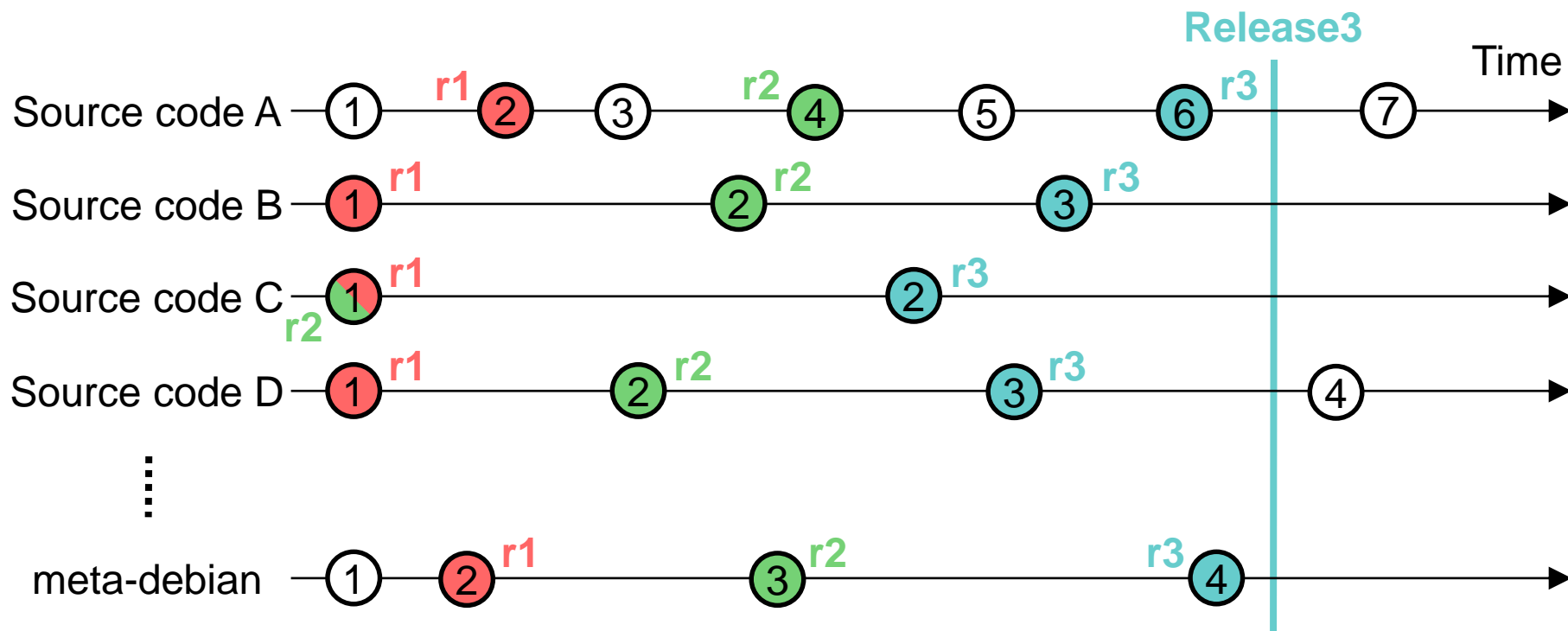


STEP1: Register a release tag



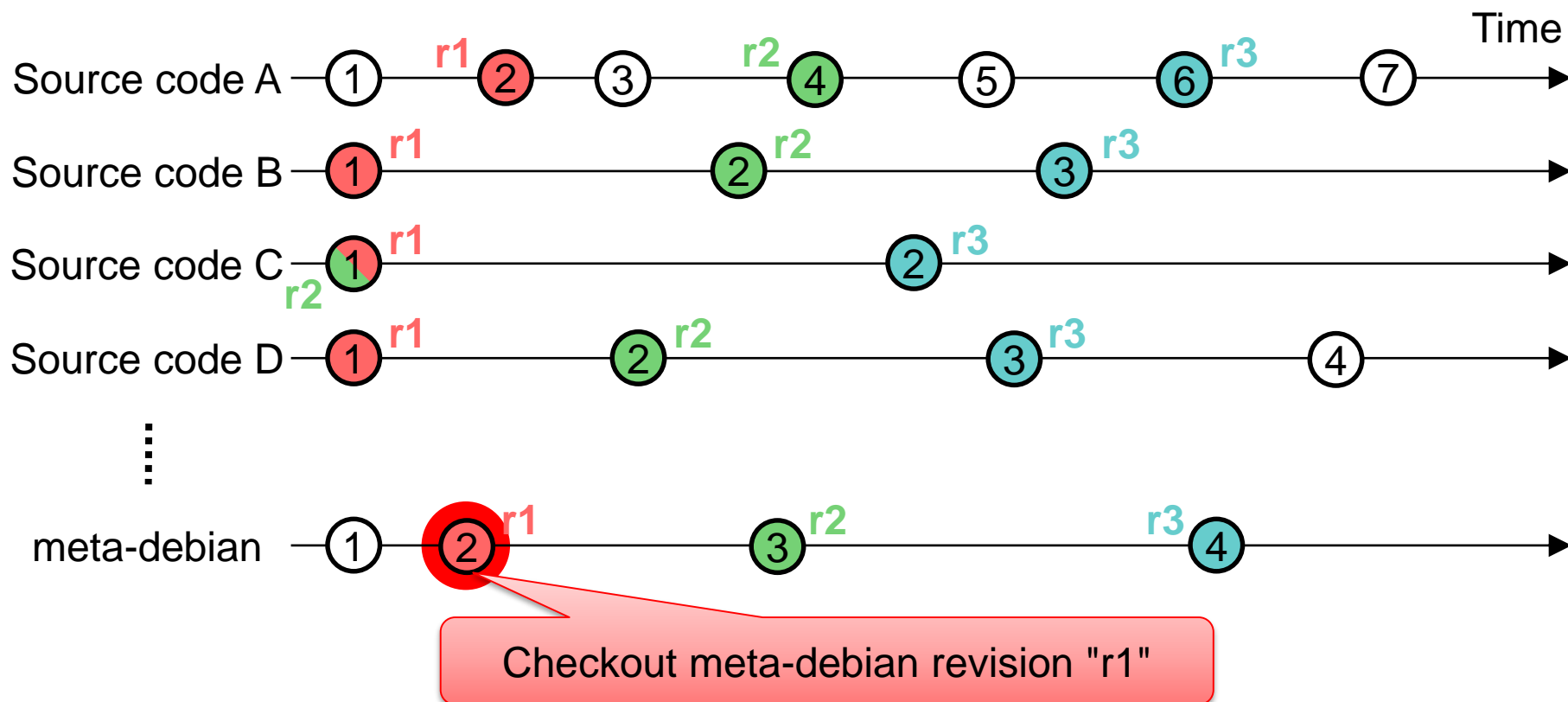


STEP1: Register a release tag



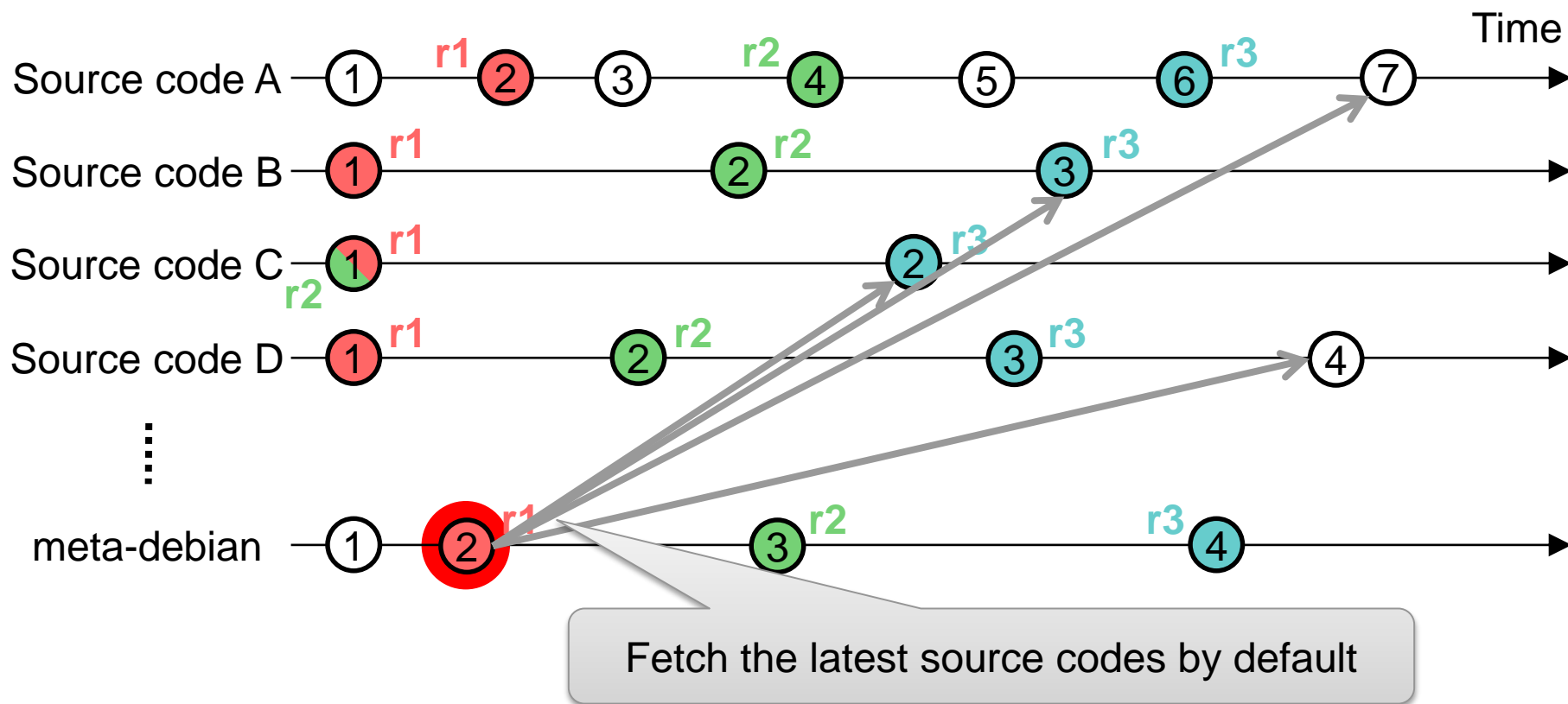


STEP2: Reproduce an old release "r1"



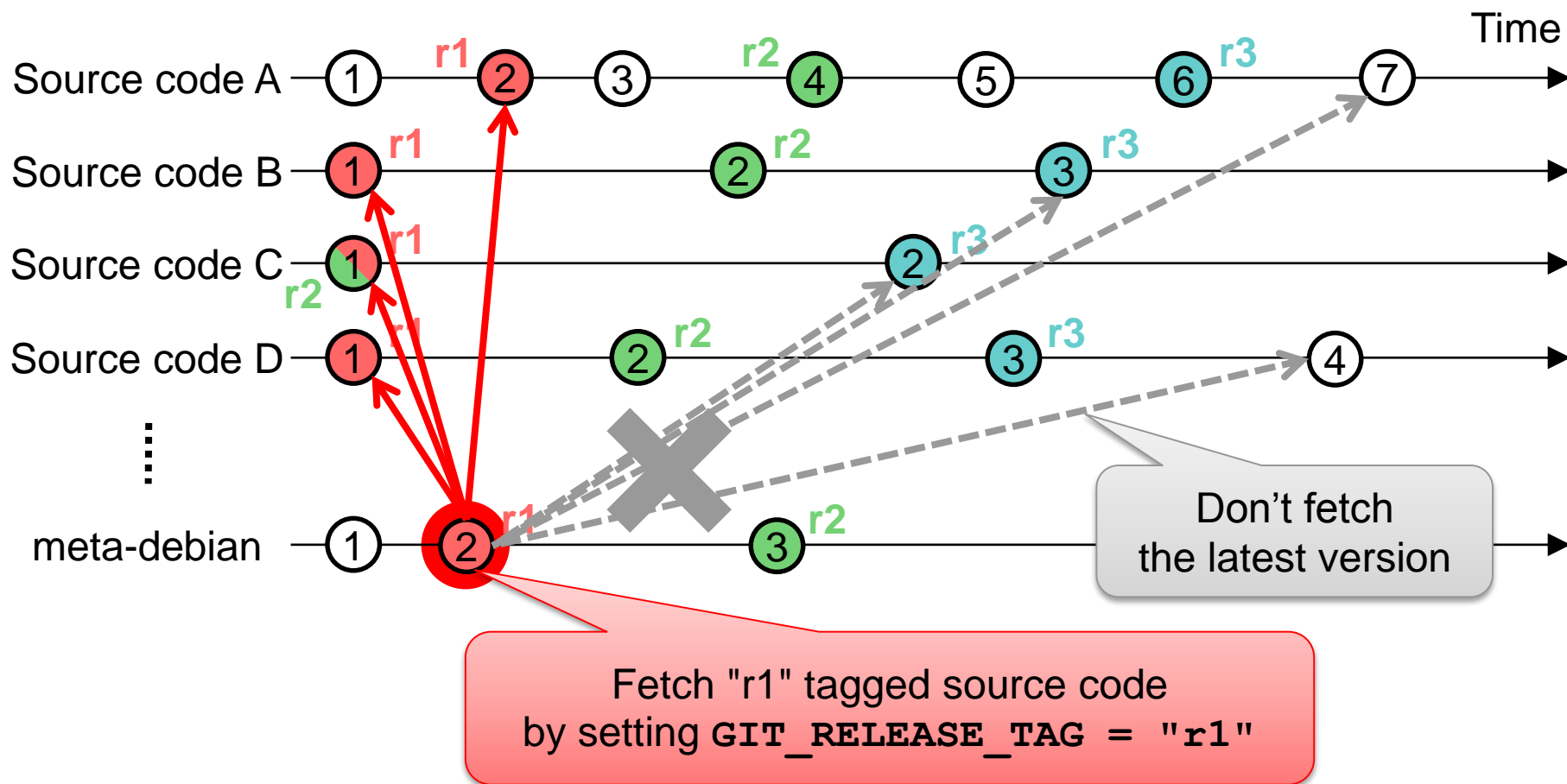


STEP2: Reproduce an old release "r1"





STEP2: Reproduce an old release "r1"

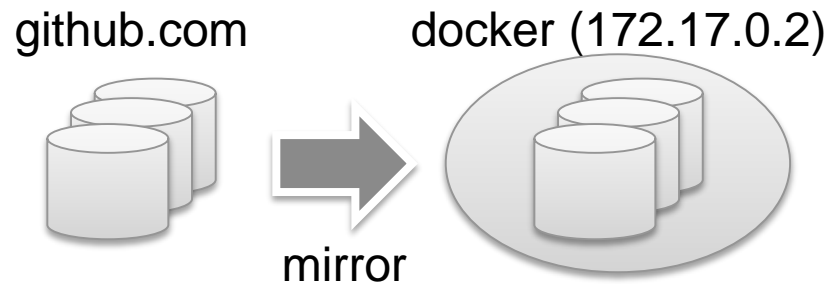




How to register tag and rebuild

- **Create git repository mirrors with docker**
 - Follow the instructions in meta-debian-docker/README.md

```
$ git clone https://github.com/meta-debian/meta-debian-docker.git
$ cd meta-debian-docker
$ ./make-docker-image.sh
$ sudo docker run -d -p 10022:22 meta-debian:1 /etc/sv/git-daemon/run -D
```



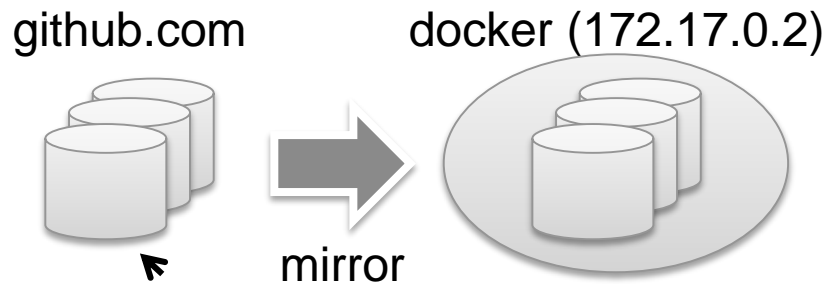


How to register tag and rebuild

- **Setup poky + meta-debian**

```
$ export TEMPLATECONF=meta-debian/conf  
$ source ./poky/oe-init-build-env
```

- **Override the git server related variables in local.conf**



Fetches source code
from github by default

poky
meta-debian

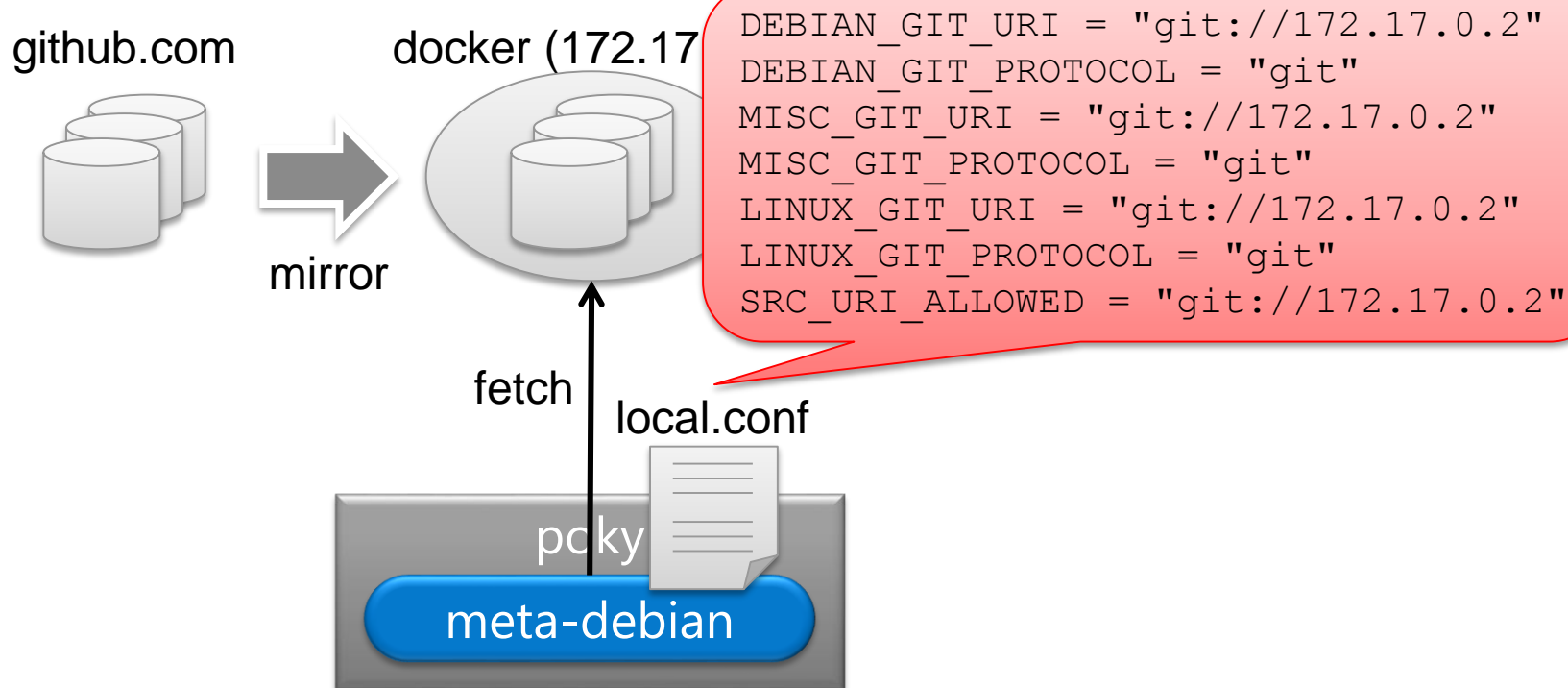


How to register tag and rebuild

- Setup poky + meta-debian

```
$ export TEMPLATECONF=meta-debian/conf  
$ source ./poky/oe-init-build-env
```

- Override the git server related variables in local.conf



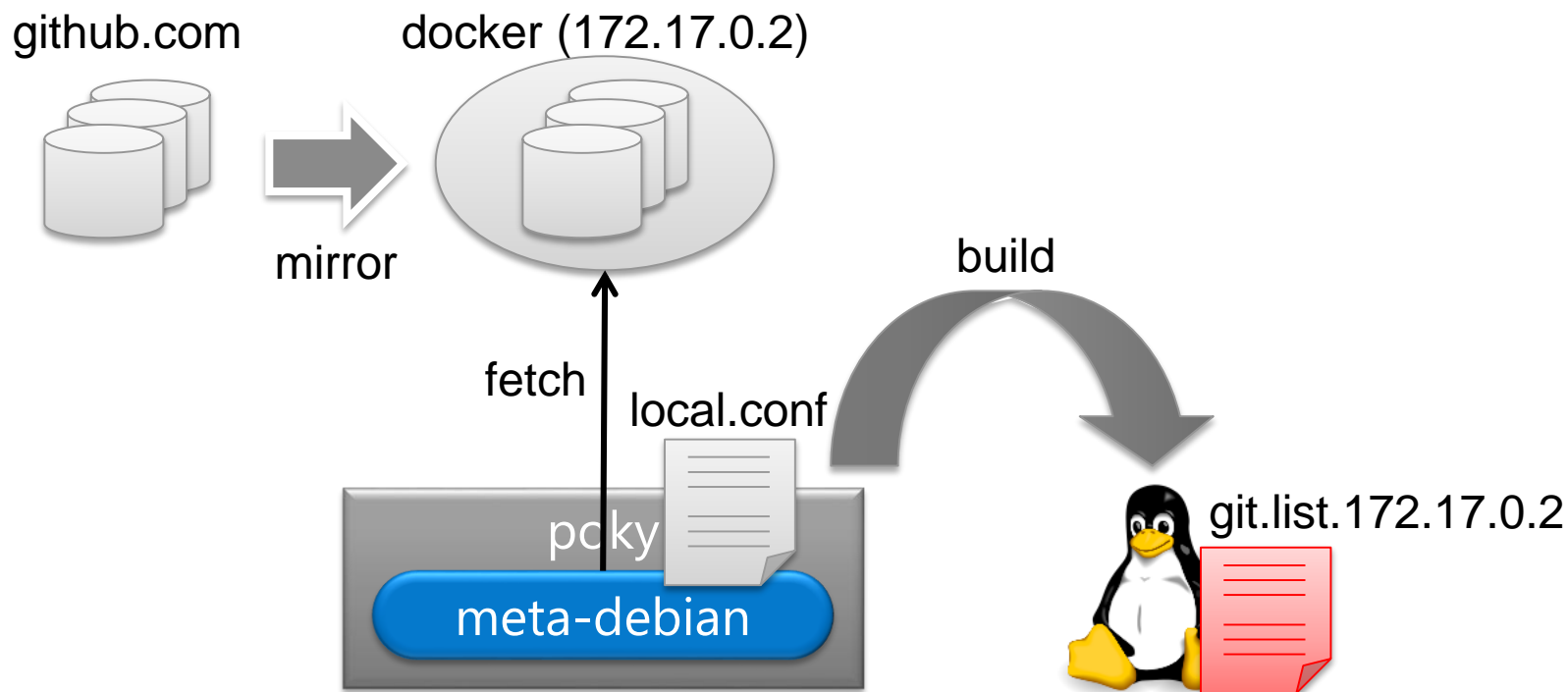


How to register tag and rebuild

- **bitbake something**

```
$ bitbake core-image-minimal
```

- **Get list files that have git repositories used in the build**
 - Example: /path/to/build/tmp/git.list.172.17.0.2

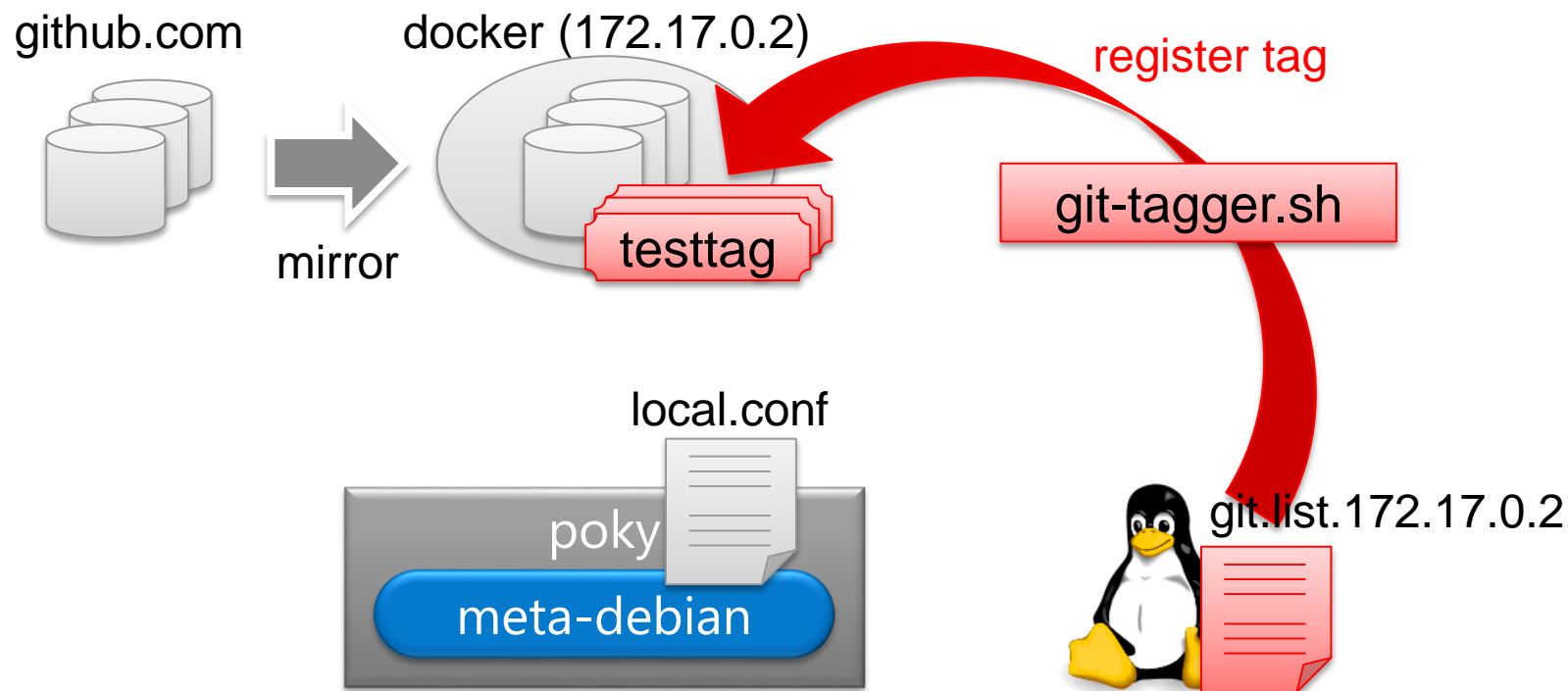




How to register tag and rebuild

- Register a tag "testtag" to the repositories

```
$ git clone https://github.com/meta-debian/meta-debian-scripts.git  
$ cd meta-debian-scripts  
$ ./git-tagger.sh git.list.172.17.0.2 172.17.0.2 testtag
```

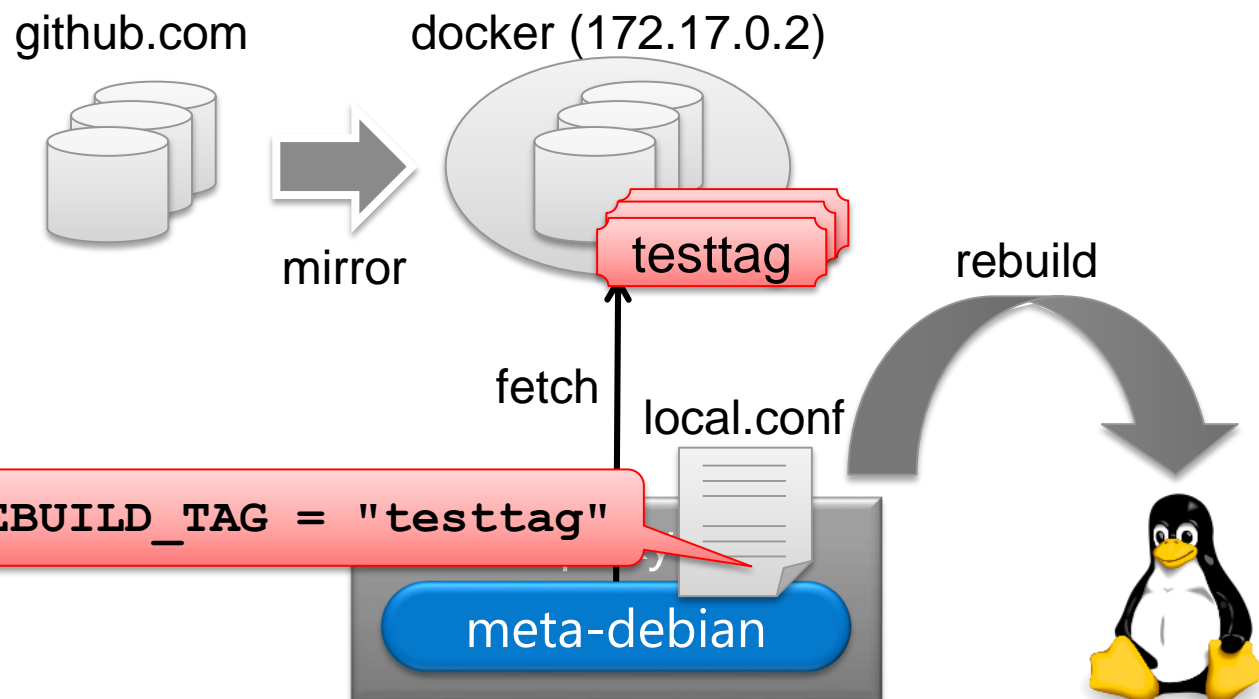




How to register tag and rebuild

- Rebuild the old image

```
$ export TEMPLATECONF=meta-debian/conf
$ source ./poky/oe-init-build-env
$ echo 'GIT_REBUILD_TAG = "testtag"' >> conf/local.conf
$ bitbake core-image-minimal
```





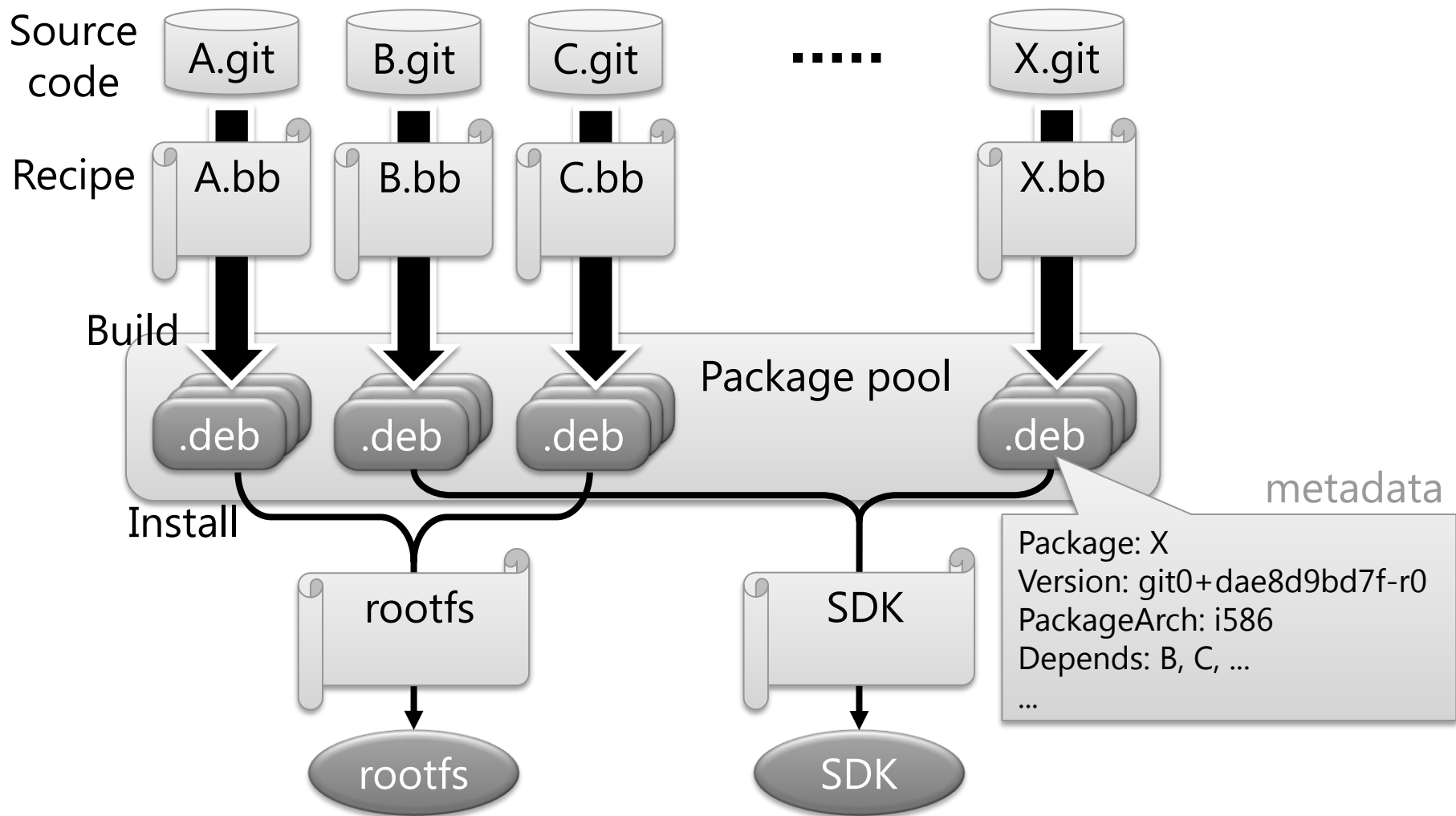
Summary generation

- **Summary information of OSS is required for products**
 - List of installed software
 - Version of each software
 - Source URI where the source code fetched
 - License of each software
- **Issues of the default poky and meta-debian**
 - Generate only a list of installed software in rootfs and SDK
- **Solution**
 - Add functions (hooks) to automatically generate summary information into rootfs and SDK recipes



Summary generation

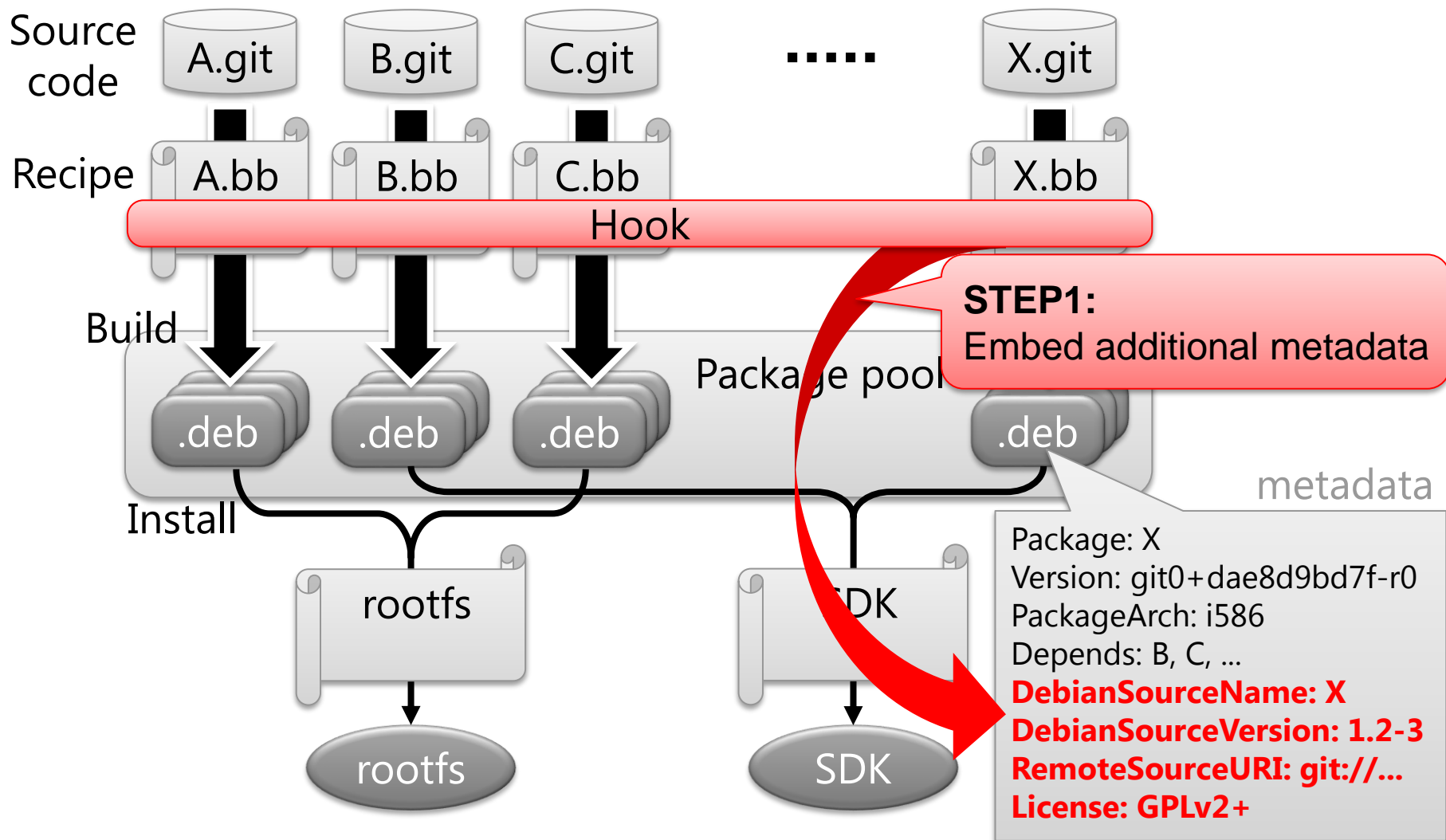
- Poky's build flow**





Summary generation

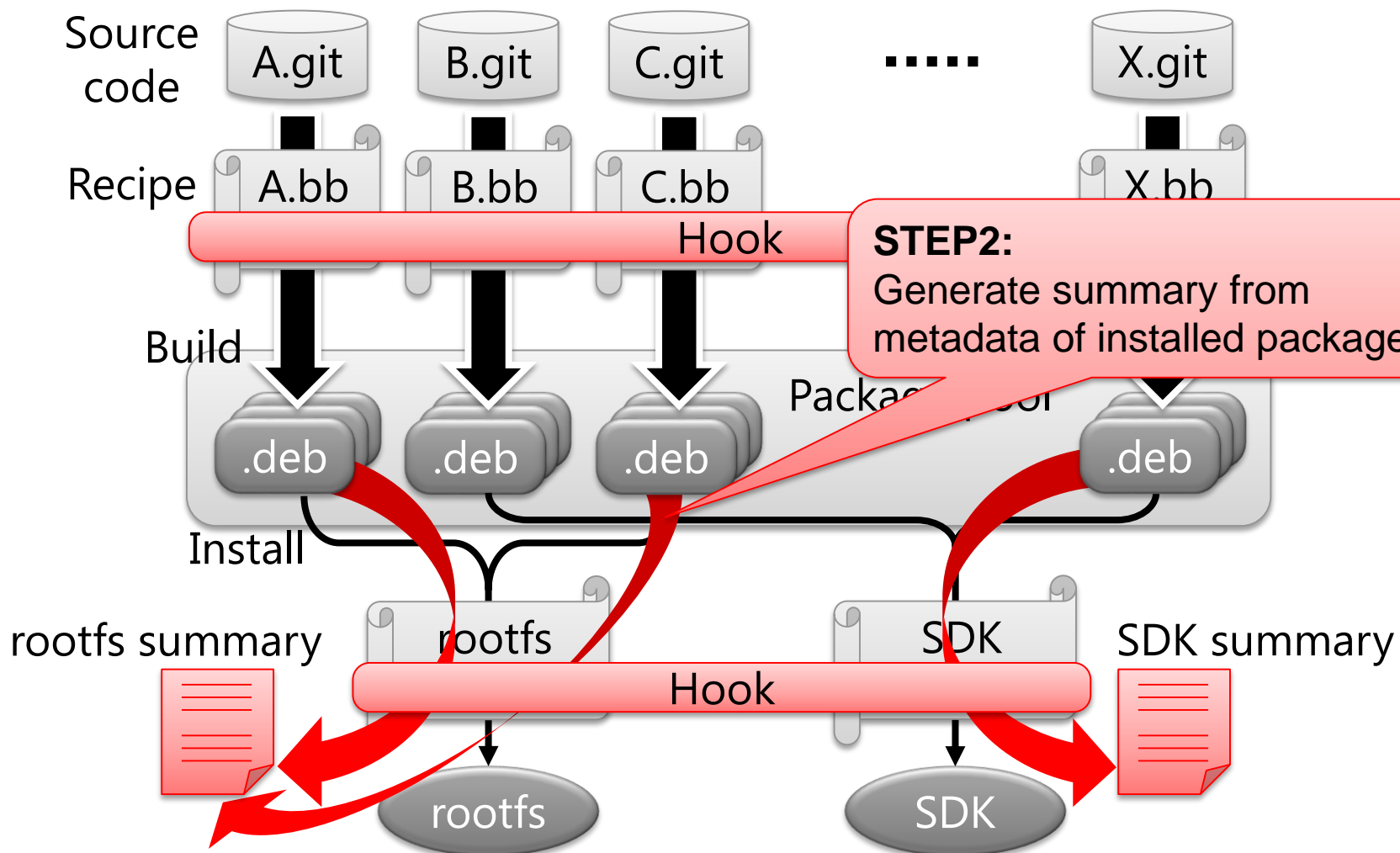
- How to collect information for each package





Summary generation

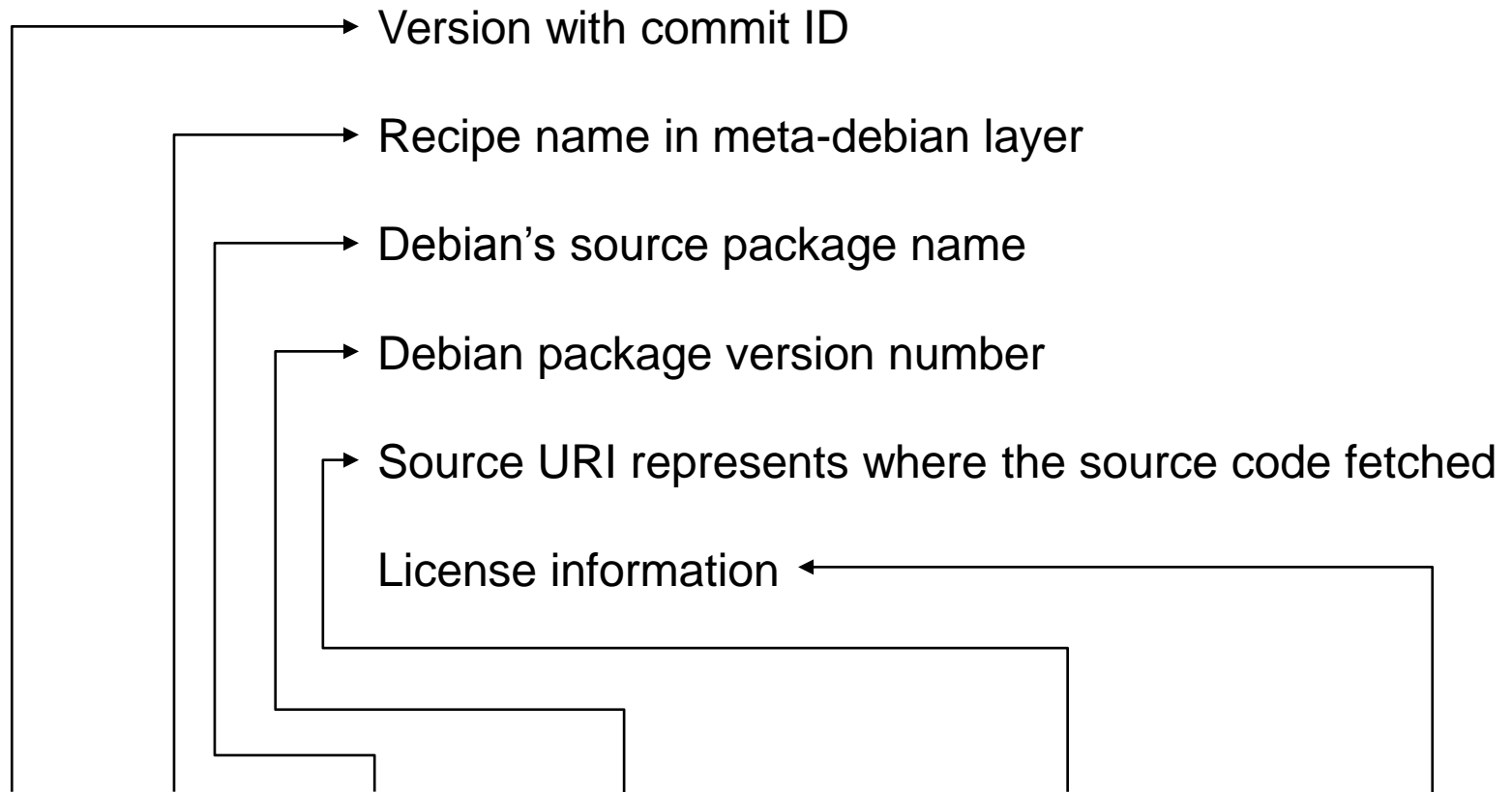
- **How to generate summary of each deployment**





Summary generation

- **Format of summary information (CSV)**



PackageName	PackageVersion	RecipeName	DebianSourceName	DebianSourceVersion	RemoteSourceURI	License
busybox	git0+8feca13beb-r0	busybox	busybox	1:1.22.0-9+deb8u1	git://localhost/busybox.git;protocol=git;branch=jessie-master	GPLv2
cpuset	git0+79474ed070-r0	cpuset	cpuset	1.5.6-4+deb8u1	git://localhost/cpuset.git;protocol=git;branch=jessie-master	GPLv2
ethtool	git0+bb474b5bf6-r0	ethtool	ethtool	1:3.16-1	git://localhost/ethtool.git;protocol=git;branch=jessie-master	GPLv2



Conclusions

- **What is Shared Embedded Linux distribution**
 - Share the work of maintaining long-term support for an embedded distribution, by leveraging the work of the Debian project
 - Metadata for building embedded Linux systems using Debian source packages
 - Implemented as an independent layer of OpenEmbedded-Core
- **Deby is intended to provide**
 - Wide embedded CPU support
 - Stability
 - Long-term support
 - Fully customizable Linux



Conclusions

- **Several new features**
 - Package management
 - dpkg / apt
 - Dynamically install/upgrade/uninstall packages at the run-time
 - Tag based source code fetch and build
 - Reproduce an old release image by setting GIT_REBUILD_TAG
 - Summary generation
 - Automatically generate summary information of rootfs and SDK



Current development status

Debian version	8 jessie (the latest stable)
Yocto Project version	2.0 jethro (stable) 2.2 morty (development)
Kernel	4.4 LTS 4.1 LTSI
BSP	QEMU: x86 (32bit, 64bit), ARM, PowerPC, MIPS VMware Player BeagleBoard PandaBoard MinnowBoard Raspberry Pi 1/2 Intel Edison board
init manager	busybox, systemd
Packages	Approx. 500



Future works

- **Keep following updates of poky and Debian**
 - Yocto Project 2.2 will be released soon (Oct. 28, 2016)
- **Support more embedded boards**
- **Improve build time for upgrading target images**
 - Related work (Binary package based approaches)
 - Isar (<https://github.com/ilbers/isar>)
 - ELBE (<http://elbe-rfs.org/>)
 - Smart Package Manager (<https://github.com/ubinux/smart2>)
- **Efficient recipe creation**
 - Add a (semi-)automated recipe generator from debian/rules
- **Integrate with LTSI test environment (Fuego)**



Please give us feedback

- **E-mail**

- yoshitake.kobayashi@toshiba.co.jp
- kazuhiro3.hayashi@toshiba.co.jp

- **Repository**

- <https://github.com/meta-debian/meta-debian.git>



Questions?
